Assignment 2

This is a group assignment. This assignment is marked out of 55 points, plus an additional 20 points towards peer evaluations.

Due Date

February 15th, 2019 11:59 PM (Final Deadline)

January 31st, 2019 11:59 PM (Project Plan Submission Deadline)

Overview of the Programming Project

The output of this programming project will serve as a starting point for your A3 (which will continue into the course project). All the programming will be done in Java/Python.

Generally, develop an interactive conversational agent that responds to user input. The agent will need to "understand" sentences typed in by the user. (This is an open research problem, so don't worry if your system doesn't seem to understand very much, or doesn't do the understanding in an "intelligent" way. Find a way to develop an agent that *mimics* understanding, such as pattern matching.)

In response to the user, the agent will need to generate sentences as output. (Consider using a wide variety of "canned" responses with variable substitutions to keep the scope of the project simple and manageable.)

To give a more realistic context to the conversational agent, you should assign it a "role" so that the agent speak according to that role. Example roles may include:

- a call centre agent
- a psychiatrist
- a celebrity
- a friend

In your conversation, you may pick a specific topic to focus on, in order to limit the scope of the language understanding part. Example of topics to use include:

- product satisfaction, compliants, reviews
- depression, stress, loneliness
- favourite food, hobbies, books

- sports

For example, if your agent is a psychiatrist, the user testing the program can act as a patient, and their dialogue can focus on a topic such as depression.

Requirements for this Assignment

For this assignment, you will focus on getting a basic agent setup, and dividing up the project into workable sizes among your team members. Specifically:

- Create a GitHub account for each of the team members (you should have done this as part of the lab).
- Create a team repository on GitHub (out of one of your team members' account) and submit that link. Your team will work out of this repository (you should have done this as part of the lab).
- Choose a software development cycle that is appropriate for this project and for your team. Think about how your team operates -- how much structure it likes and needs, etc. Give a brief rationale as to why you chose this SDLC rather than others.
- Create a project plan and submit this document. This is meant to help you do some upfront design, role assignment, task breakdown, and hours monitoring. Specifically, the report will include:
    - A brief description of your project (i.e., one paragraph), what you are doing, what role the agent and user takes on in a conversation. The URL to your team repository should be included here.
    - Your chosen SDLC and rationale for its suitability (max. one paragraph).
    - In point form: A listing of all the phases of the SDLC, where each phase should have at least 3 tasks. Here, make sure at least two of the tasks have subtasks. Give your phases, tasks, and subtasks meaningful labels -- run your labels by the TA if you're not sure they are easy to understand by a third party.
    - A Work Break Down Structure (WBS) showing task assignment to each team member, estimated duration of the task in hours (round to nearest half hour), and actual duration of task in hours (round to nearest half hour). Provide a brief explanation of your WBS.
    - A Gantt chart of your tasks, showing start and end dates for each task, and showing dependencies across tasks. Provide a brief explanation of your chart.
    - Don't forget that this assignment is not just about coding. So, when you develop your project plan, role assignment, and hour estimations/actuals,

    - be sure to include tasks such as project meetings, planning documentation, WBS creation, code documentation, etc.
    - Make a list of limitations of your program (e.g., it cannot handle synonymns, it cannot handle incorrect spelling, etc.). This list will serve as possible features for development in the next assignment.
    - Include sample output in your project report. Have one dialogue (at least 30 turns) that show a good or feasible conversation. Have at least two short dialogues that show when your agent is not able to handle the conversation properly.
- Your resulting program should be able to carry on a dialogue with the user for over 30 turns. (For simplicity, a "turn" is an instance of a prompt and a response together, where each prompt and response may be one or more sentences.)
- A README file included in the repository to explain what the project is about, how to compile and run the code.
- Generally, your code is expected to be clean, well-organized, and well documented. Be sure that:
    - The README file describes how your classes are organized. If you need to include a diagram to illustrate your class organization, use a README.doc or README.pdf rather than a txt file.
    - Each class explains what its main responsibility is. Recall: each class should only have one responsibility.
    - Methods should be short. Anything that may appear complicated in a method needs to be documented. How to judge this? If someone else on your team can't remember what a method you wrote does just by looking at the code for one minute, you need to document it.
    - Use good naming conventions so that variables, methods, and class names are easy to follow.

Design Considerations: Before you start coding, think about how your agent is going to determine what it should say next. For example, let's say the agent has 10 canned sentences in its repertoire, and some reasonable set of vocabulary. After a user enters "Hi", what will the agent output? How will it decide which of the 10 sentences to output? Next, after a user enters "What is your favourite sport?", what will the agent output? How is this decision different from the one responding to a greeting? Think about different things that a user may say in a conversation, and consider how your agent may respond to them the same or differently in each case.

Project Presentation

During the class presentation for this assignment, give a short presentation on what you have done. Specifically:

- Give a live demo of your project (about 3-4 minutes). Highlight anything that works particularly well and anything particularly poorly.
- Give a brief overview of who did what (about 1 minute).
- Show us the PM graphs of your project repository on GitHub (about 1 minute). Graphs to show: the branching structure of the code, the commits per team member

Peer Evaluations

Both A1 and A2 involved team work. As such, we want to increase awareness of the importance of team work, your own understanding of how well you work in a team, and your own understanding of what works well for you in a team environment and what doesn't. Sometimes, even if it seems frustrating, there are still good points to focus on. The purpose of these peer evaluations is to help you recognize all these points.

- You will submit peer-evaluations online (I will post the links)
- Be objective in your evaluation. Where appropriate, explain why a certain score is particularly high or low.
- Remember to point out both strengths and points for improvement. These evaluations are to help you increase your own ability for critiquing as well as to help others improve.

Evaluation Criteria

Specific evaluation criteria are:

- **5 points:** Creation of individual GitHub accounts and having a team repository there.
- **4 points:** Brief description of project being done. Description and rationale for the chosen SDLC. List of limitations of the program submitted.
- **6 points:** Listing of phases, tasks, subtasks. Breadth of coverage. Meaningful breakdown.
- **10 points:** WBS including all task assignments, hour estimations, actual hours. Distribution of workload should be fair.
- **5 points:** Gantt chart with start/end dates and dependencies.

- **10 points:** Software's ability to undergo 30 turns of a dialogue. Quality of dialogue: is the conversation "smooth" and realistic? Coverage of test output provided.
- **10 points:** Software design (at the class level) and understandability of code documentation.
- **5 points:** Presentation: does the demo work? Were branches used per feature implemented? Did everyone contribute to the project in some way?
- **5 points each round:** Four rounds of peer evaluations for each of your team member. At each round, 4 points are allocated towards your ability to critique and 1 point based on your peer reports on you.

What to Submit

The project report in PDF (submit on Canvas).

Be sure to include your **full names** in your project submission.